Information Sciences 718 (2025) 122338

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Towards heterogeneous federated graph learning via structural entropy and prototype aggregation

Zhehao Dai [a,b], Guojiang Shen [a,b], Haopeng Yuan [a,b], Shangfei Zheng [c], Yuyue Hu [a,b], Jiaxin Du [a,b], Xiangjie Kong [a,b,*], Feng Xia [d]

[a] *Zhejiang University of Technology, Hangzhou 310023, China*
[b] *Zhejiang Key Laboratory of Visual Information Intelligent Processing, Hangzhou 310023, China*
[c] *Zhejiang Sci-Tech University, Hangzhou 310018, China*
[d] *RMIT University, Melbourne, Australia*

ARTICLE INFO

Keywords:
Federated graph learning
Heterogeneity
Structural entropy
Prototype learning

ABSTRACT

In today's data-driven landscape, Federated Graph Learning (FGL) facilitates collaborative training between distributed data while providing robust privacy protections. However, FGL faces significant challenges in practical application: data heterogeneity owing to divergent node distributions and graph structures across clients, coupled with model heterogeneity caused by heterogeneous GNN architectures, substantially impedes the aggregation efficacy and generalization capabilities of global models. Existing FGL frameworks often overlook the unique impact of graph topology, inherent to graph data, between data and model heterogeneity. We propose an innovative framework called Structural Entropy Federated Graph Learning (SEFGL) that leverages structural entropy to simultaneously address data and model heterogeneity. At the client level, structural entropy-based virtual node generation and graph reconstruction techniques are applied to strengthen minority class node representations and optimize local graph topology while maintaining the original data distribution. At the server level, a prototype learning approach based on structural entropy aggregates data from clients with similar entropy characteristics. This enables each client to acquire a more diverse global representation, fostering the development of a personalized and robust prototype. Experiments conducted on three graph datasets demonstrate that the SEFGL framework achieves superior performance in terms of generalizability, efficiency, and effectiveness in high-heterogeneity scenarios.

1. Introduction

In recent years, with the rapid advancement of information technology and data science, graph data has become indispensable across various fields, including social networks, transportation networks, bioinformatics, and recommendation systems [1]. These fields primarily rely on in-depth analysis of graph data to uncover complex relationships and patterns between data points. Graph Neural Networks (GNNs) [2] have demonstrated immense potential as a powerful tool for analyzing and handling graph data. However, as awareness of data privacy and security has grown, traditional centralized data processing methods have become increasingly exposed to significant risks of privacy breaches. Federated Learning (FL) [3] has emerged as a new paradigm in distributed machine

* Corresponding author.
E-mail addresses: zhehaodai@outlook.com (Z. Dai), gjshen1975@zjut.edu.cn (G. Shen), 201806051418@zjut.edu.cn (H. Yuan), zsf@zstu.edu.cn (S. Zheng), Huyuyue0417@outlook.com (Y. Hu), jiaxin.joyce.du@gmail.com (J. Du), xjkong@ieee.org (X. Kong), f.xia@ieee.org (F. Xia).

https://doi.org/10.1016/j.ins.2025.122338
Received 24 February 2025; Received in revised form 21 May 2025; Accepted 21 May 2025

Available online 27 May 2025

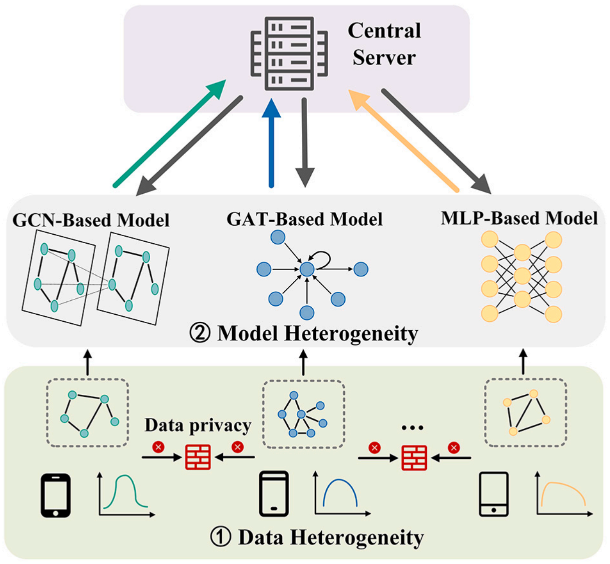


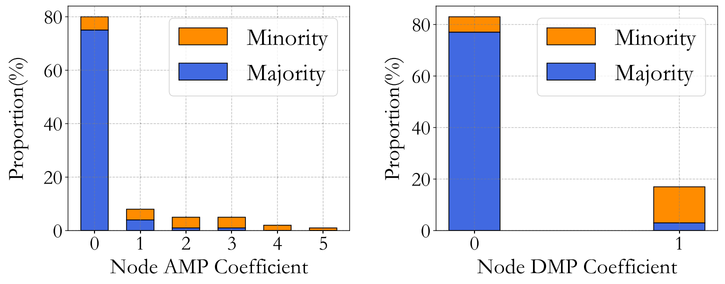
Fig. 1. Problem illustration of heterogeneous federated graph learning. 1) In the presence of data heterogeneity, what strategies can be employed to tackle the issue of imbalanced data distribution to enhance the model's generalizability? 2) In the case of model heterogeneity, what methods can effectively reduce aggregation and training issues, allowing clients with different models to access shared server data while minimizing communication overhead?

learning to address these issues. It enables multiple participants to collaboratively train a global model without centralizing or directly exchanging their data. Within this framework, Federated Graph Learning (FGL) [4], which applies FL to GNNs, offers a method for protecting the privacy of distributed graph data and models. FGL trains GNNs locally on devices by sharing only model parameters or updates, rather than raw data, thereby ensuring users' data privacy and security [5]. Despite the privacy solutions offered by FGL, several fundamental challenges remain in its practical application (Fig. 1).

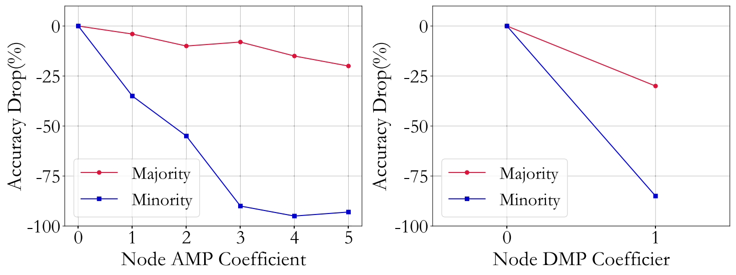
Data heterogeneity [6] refers to marked differences in data distribution across clients, particularly in node categories and graph structure. This heterogeneity is common in real-world scenarios and often manifests as class and quantity imbalance. It is also called the data imbalance phenomenon. In these cases, data concentrates in a few dominant categories, biasing the model and reducing its generalization ability and performance. Model heterogeneity [7] refers to clients using varying local GNN architectures, such as differences in convolution layers or attention mechanisms. Differences in hardware, computational power, and data characteristics lead clients to adopt distinct GNN structures, which can reduce the stability and generalization of the global model. When model structures differ substantially, training FGL becomes significantly more challenging. Subsequent experimental results demonstrate that model accuracy decreased by 9.64% with enhanced data heterogeneity, whereas the accuracy degradation reached 13.11% under increased model heterogeneity. These findings confirm that data and model heterogeneities not only exert significant adverse effects on FGL performance but also highlight the need for improving the robustness and adaptability of existing methodologies. In real-world scenarios, such as in edge computing and financial transactions, data and model heterogeneities often coexist. This dual heterogeneity significantly increases the complexity of model updates and aggregation, leading to challenges in global model aggregation and generalization.

Data heterogeneity encompasses not only distribution differences but also inherent biases in graph structures. This is evident in Adversarial Message Passing (AMP) and insufficient Distant Message Passing (DMP) [8]. AMP occurs when nodes receive information from neighbors of different categories, creating inconsistent signals that lower the Signal-to-Noise Ratio (SNR) and heighten classification error risk. Conversely, DMP refers to weak links between nodes of the same category, restricting their access to relevant supervisory signals and disproportionately impacting minority categories. Fig. 2 shows the effects of AMP and DMP on node distribution and prediction accuracy, indicating that minority nodes concentrated in high AMP/DMP regions and became more susceptible to data imbalance and noise. Federated Personalized Learning (FPL) [9] partially addresses data heterogeneity via personalized local training but incurs high training overhead and communication costs. Structural entropy [10], which reflects graph structure complexity, offers useful guidance for graph optimization. Reducing structural entropy generates an optimized graph that preserves key classification information while reducing redundancy and noise. Therefore, we employed structural entropy-based graph learning to handle data heterogeneity and improve local graph structures on clients.

Model heterogeneity [7] is a significant challenge in FGL. Clients often use varied GNN architectures owing to differences in hardware resources, computational capacities, and data characteristics. For instance, devices such as smartphones, edge servers, and high-performance computing clusters often have varying computational capabilities, leading to the adoption of different GNN architectures. It not only affects the aggregation and alignment of the global model but also exacerbates the issue of domain shift [9], where the node representations from each client's private GNN differ significantly across clients. Directly averaging or aggregating these models can lead to reduced generalization or even training failure. A prototype-based heterogeneous federated framework addresses this by reducing communication costs and ensuring precise data transmission between clients. In contrast to traditional



(a) Distribution of node



(b) Predictive performance

Fig. 2. Distribution of data nodes with AMP/DMP coefficients and the effect of AMP/DMP coefficients on node prediction accuracy.

FL, which aggregates parameters or gradients, prototype learning transmits node embeddings, boosting both privacy protection and efficiency in FL.

Traditional FL algorithms, particularly those applied in the image and text domains [11], face challenges in FGL. Graph data structures are more complex and variable, with significantly higher connectivity and feature diversity among nodes than in image and text data. Furthermore, the stringent privacy requirements for graph data and the pronounced model heterogeneity complicate model aggregation and training. Currently, few methods in the FGL domain effectively address both data and model heterogeneity simultaneously [12], underscoring the urgent need for specialized algorithms to address these unique FGL challenges.

To address these challenges, we introduce a heterogeneous FGL framework, Structural Entropy Federated Graph Learning (SEFGL), based on structural entropy [10] and prototype learning [11]. On the client side, SEFGL uses virtual node generation to enhance semantic alignment for limited-sample nodes, thereby improving classification accuracy. In addition, SEFGL applies structural entropy optimization to derive an optimal encoding tree, achieving structure-level alignment and mitigating data heterogeneity issues. On the server side, SEFGL employs structural entropy-based prototype learning. In contrast to traditional global prototype generation, SEFGL clusters clients by structural entropy similarity, selecting the most relevant class prototypes as personalized prototypes. Our main contributions can be summarized as follows:

- We propose SEFGL, an innovative FGL framework with remarkable adaptability and accuracy in handling imbalanced data distribution and heterogeneity, supporting fully heterogeneous models.
- We introduce structural entropy to integrate the entire framework. Clients enhance minority class representation by generating virtual nodes and optimizing structural entropy, thereby improving the model's generalization in data-heterogeneous environments. On the server side, client data are clustered according to structural entropy similarity, facilitating the learning of diverse global representations and yielding more robust prototypes.
- Comprehensive experiments in three benchmark datasets validate the effectiveness of SEFGL, showing significant performance improvements over baseline methods in high-heterogeneity scenarios, with superior model generalization and learning efficiency.

2. Related work

2.1. *Heterogeneous federated learning*

In real-world scenarios, heterogeneous federated learning faces several challenges, the most prominent being data heterogeneity and model heterogeneity. To address data heterogeneity, existing research [7] emphasizes the effects of non-IID data on model performance and the issue of data imbalance. To alleviate these issues, researchers have introduced various techniques, such as data augmentation, selection, and reweighting while enhancing the robustness and generalization of the global model. Model heterogeneity refers to the differences in hardware, software, and computational power across clients, which may lead to the use of different model structures or sizes. To overcome this, designing flexible FL frameworks that support the co-training of multiple models is essential. Current mainstream approaches can be classified into two categories: knowledge distillation and prototype learning.

In knowledge distillation approaches, such as FedKD [13], the challenge is addressed through parallel training of a global model and a personalized model for each client. The global model is collaboratively updated across clients, whereas the personalized model is adapted to local data distributions. Although this method is effective in mitigating data heterogeneity and reducing communication costs, it requires proxy datasets on the server side for knowledge transfer, which may not be feasible in real-world scenarios.

In contrast, prototype learning approaches, such as FedProto [11], improve global model performance by generating class prototypes to represent local data features and leveraging these prototypes for aggregation and comparison during federated training. Compared to knowledge distillation, prototype learning reduces the need to transmit large volumes of model parameters, lowering communication costs while more effectively preserving client-side data characteristics.

In recent years, several studies have explored the impact of graph structure on FGL. For instance, FedSpray [14] and FedSSP [15] have investigated how different graph structures influence the performance and convergence of FGL models. Additionally, FedVN [16] has introduced the concept of virtual nodes to address distribution shifts in FGL scenarios. These works have provided valuable insights into understanding the complexities of FGL and have inspired our research in developing more robust and adaptable FGL frameworks.

2.2. Imbalanced learning on graphs

As graph representation learning has advanced, particularly with the emergence of GNNs, novel opportunities for graph analysis have emerged. However, issues related to imbalanced graph data have become increasingly prominent [12], covering both class imbalance [17] (such as the uneven distribution of labels across nodes, edges, and graphs) and structural imbalance [18], including variations in node degrees, topological structures, and graph sizes. To mitigate these challenges, researchers have introduced diverse methods, including algorithm-level approaches [19,20] and data-level techniques [21] for imbalanced node classification. In few-shot tasks, such as node, link, and graph classification and reasoning, meta-learning [22] and data augmentation [20] have been applied. To address structural imbalances, techniques such as knowledge transfer [18], and public knowledge sharing—and the use of auxiliary data are applied. Methods such as data reweighting, sampling, synthetic data generation, and constraint integration are employed to balance high- and low-resource components.

2.3. Structural entropy

In recent years, structural entropy based on coding trees has been introduced as a measure of information in graph structures [10], aiding in the discovery of natural hierarchies embedded within graphs. Structural entropy has emerged as a significant research direction in GNNs. In SEGA [23], the authors present a novel framework for graph comparison learning that leverages structural entropy to guide the creation of anchored graph views, leading to substantial improvements in various graph classification tasks. SEP [24], a hierarchical pooling method, is introduced to decode the hierarchical structure of a graph by minimizing structural entropy, preserving local structures while eliminating noisy elements. In SE-GSL [25], the authors propose a general framework for graph structure learning that optimizes graph structures using structural entropy and hierarchical encoding trees. Although these works explore structural entropy extensively, its application in FGL remains underexplored, presenting a promising avenue for further research.

2.4. Graph reconstruction

Graph reconstruction has become crucial for optimizing graph representations in clustering tasks. AONGR [26] developed an auto-weighted multiview graph reconstruction method that enforces orthogonal and nonnegative constraints to fuse complementary information while learning view-specific weights. EGRC-Net [27], which jointly optimizes node embeddings and adjacency matrices through feature-aware proximity, enables iterative graph refinement in deep clustering frameworks. For ensemble scenarios, Auto-weighted Graph Reconstruction (ARG) [28] unified heterogeneous clustering results into a consensus graph via adaptive weighting, balancing efficiency and structural preservation. These methods highlight that strategic graph reconstruction—whether through multiview fusion, embedding guidance, or ensemble integration—significantly enhances clustering performance. Our work extends these principles to federated scenarios, replacing weight learning with structural entropy optimization to refine graph representations for local clients.

2.5. Prototype learning

Prototype learning [29] is an instance-based machine learning method, where prototypes are generated by averaging the representations of all samples within each category, capturing the representative features of each category to classify new data. In prototype learning frameworks, only prototype data are transmitted without sharing model parameters. As low-dimensional representations, prototypes exhibit an irreversible generation process that effectively prevents inversion attacks from reconstructing raw data, thereby significantly enhancing privacy protection effectiveness. This approach demonstrates considerable potential in practical applications. For instance, in recommender systems, prototype learning effectively identifies user preference patterns to provide personalized recommendations [30]. In natural language processing, it enhances the learning ability of pre-trained models with fewer samples, improving text categorization accuracy [31]. Additionally, in FL, algorithms such as FedProto [11] and FedTGP [32] significantly enhance model performance by utilizing prototype data as shared global knowledge during client updates. Owing to its advantages in computational simplicity and low communication cost, prototype learning holds promising applications in heterogeneous federated learning, which will be explored in detail in the following section.

3. Preliminary

3.1. Federated graph learning

In the context of FGL, each client manages a unique subgraph dataset G_i, with potentially significant differences in structure and features. To accommodate the specific characteristics of their respective datasets, clients may adopt distinct GNN [33,34] architectures.

Thus, the GNN model of the i-th client can be expressed as $f_i(\theta_i)$, where θ_i represents the model's weight parameters, which constitute the private information of the client i. During the federated learning process, the objective for all clients is to collaboratively train a global model that integrates local knowledge from each client while optimizing the following objective function:

$$L = \sum_{n=1}^{N} \alpha_n L_n(f_n(\theta_n), G_n), \tag{1}$$

where L_n denotes the local loss function of the n-th client, which evaluates the performance of model f_n on its subgraph G_n. α_n represents a weight coefficient that adjusts the relative importance of each client in the global objective function.

3.2. AMP/DMP-sourced bias

The AMP coefficient, $amp_k(u)$, is defined as the ratio of non-category nodes to category nodes within the k-hop neighborhood of node u:

$$amp_k(u_i) := \frac{|\{v | v \notin V_i, v \in H(u_i, k)\}|}{|\{v | v \in V_i, v \in H(u_i, k)\}|}, \tag{2}$$

where V_i is the set of nodes in category i, and $H(u_i, k)$ represents the set of nodes in the k-hop neighborhood of node u_i.

The DMP coefficient, $dmp_k(u)$, is defined as an indicator function, where if all labeled nodes in the k-hop neighborhood of node u do not belong to the same category as u, the function is equal to:

$$dmp_k(u) := 1(L_{ik}(u) = 0, \sum_j L_{jk}(u) > 0), \tag{3}$$

where $L_{jk}(u)$ denotes the number of labeled nodes in the k-hop neighborhood of node u that belong to category j.

3.3. Structural entropy

Structural entropy [10] is a technique used to quantify the uncertainty of information within a graph's structure. It reflects the complexity of node or edge distributions within the graph and is commonly employed to assess the graph's organizational structure and the complexity of information propagation.

One-dimensional structural entropy is a measure of the complexity within a single layer or community in a graph. It is typically defined as:

$$SE^{(1)}(G) = -\sum_{u \in V} p_u \log p_u, \tag{4}$$

$$p_u = \frac{\deg(u)}{\sum_{v \in V} \deg(v)}, \tag{5}$$

where $deg(u)$ is the degree of node u, which refers to the number of edges connected to node u.

High-dimensional structural entropy takes into account the hierarchical division of a graph, aiming to quantify the distribution and uncertainty of information within the graph. It is typically defined by constructing an encoding tree of the graph.

$$SE^{(k)}(G) = \min_{\forall T:\mathrm{height}(T) \le k} \{SE^T(G)\}, \tag{6}$$

$$SE^T(G) = -\sum_{u \in T, u \neq \rho} \frac{W(C_u)}{V_u} \log_2 \left(\frac{W(C_u)}{V_{u^-}} \right), \tag{7}$$

where the k-dimensional structural entropy of a graph G is $SE^{(k)}(G)$. The height of the encoding tree T is denoted as height (T), while $SE^T(G)$ represents the structural entropy corresponding to the encoding tree T. ρ denotes the root node of the encoding tree T. $W(C_u)$ denotes the sum of the weights of all edges within community C_u. V_u denotes the sum of the degrees of all vertices within community C_u, and V_{u^-} refers to the sum of the vertex degrees of node u's parent node in graph G.

The one-dimensional structural entropy measures the informational complexity of a single-layer graph structure, whereas the high-dimensional structural entropy captures hierarchical structure information by constructing a coding tree, facilitating data heterogeneity optimization. High-dimensional structural entropy also reveals deeper community structures within the graph, enabling more effective clustering and structural optimization under heterogeneous data conditions.

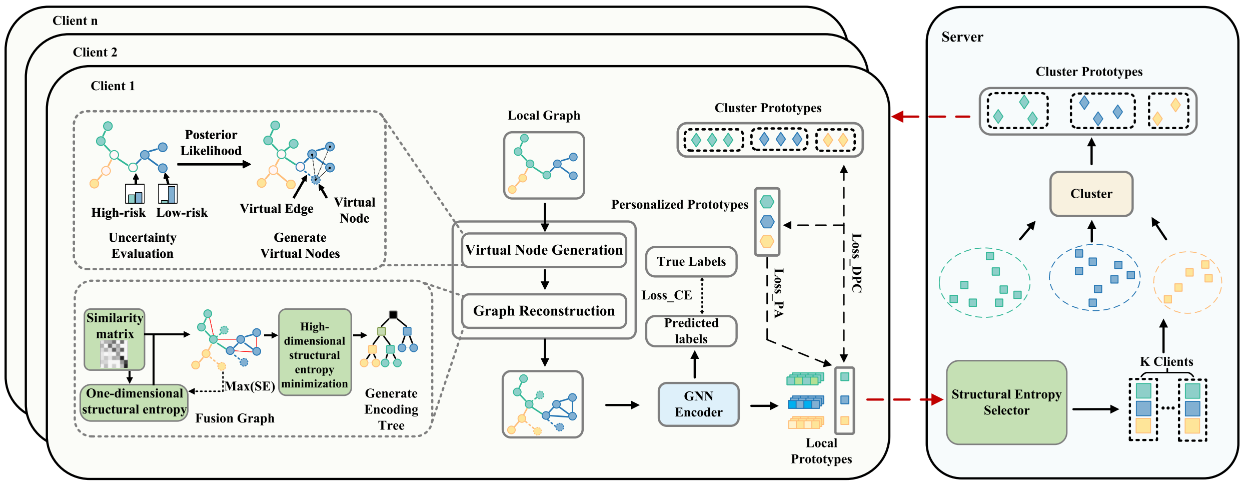


Fig. 3. The proposed SEFGL framework. The left box represents the client side, which includes the virtual node generation module, the graph reconstruction module, and the model update process. The right box represents the server side, responsible for performing the personalized prototype calculation and sending the unique prototype data of each client back to the respective client.

4. Methodology

In this section, we describe the SEFGL framework in detail. The entire framework is illustrated in Fig. 3, outlining the training process on both the server and client sides. We first enhance the graph structure of local client data through two key steps: virtual node generation and graph reconstruction. These steps implement targeted measures to address data imbalance at both the node and structural levels, effectively alleviating the challenges of data heterogeneity confronted by clients. Subsequently, structural entropy-based prototype learning is conducted on the server side to address the heterogeneity of the local client's model. The following sections offer a detailed explanation of the key technologies and implementation steps within this framework.

4.1. Local graph structure learning

Data imbalance among local clients has long posed a significant challenge in FL. Specifically, AMP and DMP are two topological phenomena that describe the class imbalance problem in graph data. AMP refers to an increase in information noise caused by a high proportion of nonsimilar or unrelated nodes within a node's local neighborhood. DMP refers to distant connections between nodes of the same or related classes, hindering the effective propagation of useful information. Both phenomena significantly impact the performance of GNNs in node classification.

We employed graph structure learning within this client to mitigate the effects of AMP and DMP on node classification and to avoid dependence on predefined graph structures. We optimized and assessed high-risk nodes in the local graph data, particularly those with a high likelihood of misclassification. The top right section of Fig. 3 illustrates the detailed process of enhancing the topology of the graph data.

Total variation (TV) distance is a statistical measure that quantifies the difference between two probability distributions, defined as the maximum discrepancy in the probabilities assigned to identical events by the two distributions. And it is employed to quantify the uncertainty $U(v)$ of node v. This is accomplished by calculating the difference between the predicted probability vector and the one-hot encoded vector of the predicted label:

$$U(v) = d_{TV}(\hat{p_v}, 1_{\hat{y}_v}) = \frac{1}{2}\sum_{i=1}^{C} |\hat{p_v}^{(i)} - 1_{\hat{y}_v}^{(i)}|, \tag{8}$$

where $\hat{y_v}$ is the predicted label of node v, and $\hat{p_v}^{(i)}$ is the predicted probability that the node belongs to category i.

After assessing the risk for each node, the local graph structure learning proceeds to estimate the posterior likelihood of the true class for high-risk nodes to correct prediction errors through topology enhancement. The posterior likelihood calculates the classification probability of a node based on its neighborhood information, which is defined as the estimated probability of its true class. The posterior likelihood estimation is divided into two main approaches:

Zero-order estimation: Based on the current model's predicted probabilities, the posterior likelihood $\hat{posterior}_v(i)$ that node v belongs to category i is computed as follows:

$$\hat{posterior}_v^{(i)} = \begin{cases} \hat{p}_v^{(i)}/(1-\hat{p}_v^{(\hat{y}_v)}), & \text{if} i \neq \hat{y}_v, \\ 0, & \text{otherwise}, \end{cases} \tag{9}$$

where $\hat{y_v}$ is the predicted label of node v, and $\hat{p_v}(i)$ represents the predicted probability that the node belongs to category i.

First-order estimation: Considering the association between node v and its neighboring nodes, the distribution of node categories in the local topology is estimated through a one-step random walk. The estimation formula is as follows:

$$\hat{posterior}_v^{(i)} = \begin{cases} \frac{|\{v' \in N(v) | \hat{y}_{v'} = i\}|}{|N(v)| - |\{v' \in N(v) | \hat{y}_{v'} = \hat{y}_v\}|}, & \text{if } i \neq \hat{y}_v, \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

where $N(v)$ represents the set of neighboring nodes of node v. These two estimation methods assist in identifying the most likely true class for high-risk nodes, laying the foundation for subsequent topology enhancement.

4.1.1. Virtual node generation

For high-risk nodes, the model enhances graph topology by introducing virtual nodes and edges. The virtual nodes represent the core characteristics of each category, whereas the virtual edges link these nodes to the high-risk nodes, delivering more accurate category information.

For each category i, a virtual node v_i^* is created, and its features are computed as the mean of the features of all nodes in that category. Based on the estimated posterior likelihood $\hat{posterior}_v(i)$, the connection probability of virtual edges is calculated as:

$$q_v^{(j)} = \omega_v \times \hat{posterior}_v^{(j)}, \tag{11}$$

After identifying high-risk nodes, virtual edges are generated using the aforementioned formula to connect virtual nodes to these high-risk nodes. This provides extensive category information for these nodes, thereby improving the classification accuracy of minority class nodes. A discount factor ω_v is introduced to adjust the connection probability of virtual edges, ensuring that only high-risk nodes requiring enhancement receive virtual connections. The discount factor is determined by the node's risk score.

The optimal discount factor ω_v is determined by solving a constrained quadratic programming problem, encouraging the creation of virtual edges for high-risk nodes while minimizing the total number of virtual edges:

$$\min_{\omega \geq 0} \left(-\sum_v (s_v - \bar{s}_{\hat{y}_v}) \omega_v + \lambda \parallel \omega \parallel_2^2 \right), \tag{12}$$

$$s_v = \frac{U_(v)}{\max_{j=1}^{J} \frac{|N_j|}{|N_{\hat{y}_v}|}}. \tag{13}$$

Here, s_v denotes the risk score of node v, N_j is the set of labeled nodes of category j, $\hat{y}_v$ is the predicted label of node v. λ is the regularization strength parameter, and $\parallel \omega \parallel_2^2$ represents the L2 norm of ω.

Traditional data augmentation methods, such as GANs [19] or GraphSMOTE [35], enhance the sample size of minority classes by generating new synthetic nodes. These nodes are considered real nodes during model training, which alters the distribution and quantity of local client data, subsequently influencing local data classification. This approach can inadvertently enhance the classification accuracy of local models. However, in the FL scenario, using traditional data augmentation methods is unfair compared to other federated approaches because these methods may provide the model with more information. In contrast, virtual nodes do not increase actual data volume and do not directly participate in the model training process. Instead, they serve as an auxiliary mechanism to assist minority class nodes in achieving more accurate classifications through graph structure adjustments. Additionally, the virtual node generation method exhibits lower complexity, rendering it computationally efficient for large-scale graph data, whereas traditional data augmentation methods can significantly raise computational costs by generating numerous new nodes.

4.1.2. Local graph reconstruction

The local graph reconstruction phase begins with edge structure enhancement, which aims to integrate vertex attributes and neighborhood information to optimize the graph's edge structure. The pairwise similarity matrix S between vertices in the graph is calculated, with Pearson Correlation Coefficients (PCCs) used as the similarity metric:

$$\rho(v_{nodeA}, v_{nodeB}) = \frac{E((v_{nodeA} - \mu_A)(v_{nodeB} - \mu_B))}{\sigma_A \sigma_B}, \tag{14}$$

where v_{nodeA} and v_{nodeB} are the feature vectors of vertices $nodeA$ and $nodeB$, respectively, μ_A and μ_B denote the means of these feature vectors. σ_A and σ_B denote the standard deviations of the feature vectors. $E(\cdot)$ represents the expected value (mean) of the product of two vector elements. Based on this similarity matrix S, a k-nearest neighbor graph $G_k = \{V, E_k\}$ is constructed, where each edge E_k represents the connection between a vertex and its k-nearest neighbors. Then, the knn graph is fusion with the original graph G to form a new graph $G' = \{V, E \cup E_k\}$.

In the k-nearest neighbor graph, the choice of the k value is crucial because an extremely low k value may lead to insufficient information and an exceedingly high k value may introduce excessive noise into G', reducing computational efficiency. To select the optimal k, we guide the choice by maximizing the one-dimensional structural entropy $SE^{(1)}(G')$. This process leverages the theory of structural entropy [10] to ensure that the information embedded in the graph is maximized, thereby enhancing the GNN's ability to learn from the graph's structural data.

Using the obtained topology and vertex similarity obtained above, the goal at this stage is to obtain the optimal coding tree. By applying the principle of high-dimensional structural entropy minimization, a hierarchical encoding tree T is generated, representing the community structure of the graph at multiple levels. The formula for high-dimensional structural entropy minimization is provided in the Related Work section.

A greedy algorithm is used to construct the optimal encoding tree T^*, which minimizes the tree's structural entropy. Each node in the encoding tree represents a community, and through combining and lifting operations, the tree's structure is dynamically adjusted to minimize the overall uncertainty of the graph. Specifically, the operation that maximally reduces structural entropy is chosen and executed between nonroot nodes and their parent nodes. This process is repeated until the encoding tree's height is less than k and structural entropy can no longer be reduced. The optimal encoding tree is represented as:

$$T^* = \underset{\forall T: height(T)\leq H}{\arg min} (SE^T(G)). \tag{15}$$

Subsequently, graph reconstruction is performed on the obtained optimal coding tree. We employed a probability sampling method based on structural entropy reduction to select edges that should be strengthened or weakened. This method effectively restores the topology of the encoding tree T^* and preserves the hierarchical semantics within the tree. The specific process involves iteratively selecting pairs of leaf nodes and calculating the probability of their connection to generate new edges. Each sampling is based on the reduction in structural entropy from the root node of the encoding tree to the current leaf node. The probability of selecting node V_i is defined as follows:

$$prob(v_n^{<i>}) = \frac{exp(SE^T(G'; (r_, v^{<i>})))}{\sum_{j=1} exp(SE^T(G'; (r_, v^{<j>})))}, \tag{16}$$

where $SE^T(G'; (r_, v^{<i>}))$ measures the uncertainty or breadth of information distribution along the path from r to $v^{<i>}$. The greater the structural entropy, the broader the information distribution along the path, and the higher the uncertainty. The process exponentiates the structural entropy values and normalizes them, expressing the probability of selecting each child node for edge construction. Node pairs with higher structural entropy are prioritized, optimizing graph reconstruction.

Structural entropy not only provides a principle for detecting the natural or true structure of graphs but also uncovers high-dimensional structural information inherent to them. This approach is particularly well-suited for FGL because it facilitates the dynamic updating of coding trees and the incremental calculation of structural entropy. It has been theoretically proven that minimizing structural entropy can generate an optimal encoding tree [36]. The optimal encoding tree can extract hierarchical community information from the graph and eliminate redundant noise, thereby improving node classification performance. Consequently, structural entropy effectively captures the dynamic evolution of communities while substantially reducing computational time. Thus, integrating structural entropy into FGL to mitigate data heterogeneity represents a sound and effective strategy.

Graph structure enhancement and graph reconstruction help mitigate data imbalance by redesigning the graph's topology to balance or emphasize important nodes and edges, enabling the model to more effectively recognize and process different data patterns.

Algorithm 1 SEFGL-Client Algorithm.

```
Input:
Client numbers $N$,
Client-side graph data $G(V, E)$.
Output:
Client-side local prototypes,
Total loss.
Local Executes:
 1: for each client numbers $n = 1$ to $N$ do
 2:     Virtual Node Generation:
 3:     $\hat{y}, \hat{P} \leftarrow F(V, E, \theta)$
 4:     $s \leftarrow CalculateRisk(\hat{y}, \hat{P})$ by Eq. (13)
 5:     # EstimatePosterior
 6:     $pos\hat{t}erior \leftarrow (G, \hat{P}, \hat{y}, s)$ by Eq. (9)-(10)
 7:     $G', V' \leftarrow AugmentTopology(G, V, \hat{p}, s)$
 8:     Graph Reconstruction:
 9:     $V'_w \leftarrow$ Edge structure enhancement
10:     # Generate the optimal Encoding Tree
11:     $T^*, SE^T(G) \leftarrow (G, V)$ by Eq. (15)
12:     $G'', V'' \leftarrow Regenerate(G', V', T^*)$
13:     # Computing Local Prototypes
14:     $P_{local} \leftarrow LocalGNN(G'', V'')$ by Eq. (17)
15:     Send $SE^T(G), P_{local}$ to Server side
16:     Run SEFGL-Server Algorithm
17:     Receive Prototype $D_i$
18:     $l_{CE}, l_{PA}, l_{DPC} \leftarrow$ Calculating total loss
19:     Update local Model by Eq. (27)
20: end for
```

4.2. Personalized prototype calculation

After completing graph structure learning and reconstruction, we introduce structural entropy to enable the personalized computation of prototypes, facilitating a more precise differentiation of local prototypes among clients. In contrast to standard prototype learning, which generates a unified global prototype, we identify client groups with similar entropy by calculating high-dimensional structural entropy variations. Because similar high-dimensional structural entropy often reflects comparable data distributions, this trait is crucial in FL because it assists in aggregating data from clients with common features. In this way, we can determine which clients have similar category distributions without direct data access, thereby supporting reliable personalized model clustering. Finally, we send the personalized clustered prototype data back to local clients, which enhances each client's model adaptability and strengthens the system's capacity to mitigate data and model heterogeneity.

Specifically, we first calculate the local category prototypes. By inputting the locally reconstructed graph data, which have undergone graph structure learning, into the local GNN model, we obtain node embeddings for each category. We then average these node embeddings to derive the local category prototypes, as shown in the following formula:

$$P_n^j = \frac{1}{|\mathcal{V}_n^j|} \sum_{v \in \mathcal{V}_n^j} E_v, \tag{17}$$

where, $\mathcal{V}_n$ represents the node set of the n-th client and E_n is the node embedding of node n.

Subsequently, we upload the local category prototypes to the server. As previously noted, directly averaging all clients' local category prototypes is overly simplistic and fails to provide differentiated guidance for clients with high model heterogeneity. To address this, we calculate the structural entropy for each client and select clients with similar entropy.

Our framework inherently ensures privacy protection without requiring additional privacy-preserving mechanisms. This is because the prototype data, serving as low-dimensional knowledge representations, is designed to exclude sensitive client information by construction. This intrinsic design fundamentally eliminates the need for supplementary privacy-enhancing operations.

We then compute the distance between the structural entropy of the target client n and that of other clients k using the following formula:

$$d_{nk} = |SE_n - SE_k|. \tag{18}$$

For each client n, we select the m clients with the smallest distance d_{nk}. This can be formally represented as:

$$M_n = \mathrm{top}_m \left(\{ P_k \mid k \in \mathrm{argmin}_m(d_{nk}) \} \right), \tag{19}$$

where M_n represents the set of the m clients closest to client n. Based on the prototype data from each client, we obtain a unique prototype dataset M for each client. The FINCH [37] clustering method is employed to cluster the category prototype data. The clustering process can be summarized as follows:

$$Q_i \xleftarrow{cluster} M_i. \tag{20}$$

Subsequently, we return the clustered category prototypes derived from the server to their respective clients. Specifically, upon receiving the personalized prototype set from the server, the client identifies the prototype in the clustered set that is closest to its local prototype. Cosine similarity is used to evaluate the distance between the client's local prototype and those in the clustered set. The prototype of the nearest cluster $Q_{i,pos}$ is determined as follows:

$$Q_{i,pos} = \arg \max_{Q_{i,j} \in Q_i} Similarity(Q_{i,j}, P_i), \tag{21}$$

$$Similarity(Q_{i,j}, P_i) = \frac{Q_{i,j} \cdot P_i}{||Q_{i,j}|| \cdot ||P_i||}. \tag{22}$$

To improve the comprehensiveness of personalized prototype data, we introduce prototype data from other categories from the same client's prototype data and use a weighted averaging method for fusion. Prototypes from categories that are more similar to the local category prototype are assigned greater weight to maintain their significant influence. The calculation process for the personalized prototype D_i of client n is as follows:

$$D_i = \lambda Q_{i,pos} + (1 - \lambda) \frac{1}{m-1} \sum_{i \neq \mathrm{k}} Q_{i,k}. \tag{23}$$

This method ensures that each client receives personalized guidance tailored to its local data and structure.

4.3. Local model update

In this section, we use the personalized prototype data received by the client to update the local model. This feedback mechanism accelerates the learning process and improves the adaptability and precision of the model when handling local data.

Algorithm 2 SEFGL-Server Algorithm.

```
Input:
Client numbers $N$,
Communication rounds $T$,
Client's local prototypes $P_j^i$,
Structural Entropy $SE$.
Output:
Personalized domain prototypes $D$.
Server Executes:
1: for each round $t = 1$ to $T$ do
2:     for each client numbers 1 to $N$ do
3:         Get $k$ clients with closest $SE$ by Eq. (18)
4:         # Get the set of clustering prototypes
5:         $Q_j^i = cluster(P_1^j ... P_k^j)$ by Eq. (20)-(22)
6:         $D_i \leftarrow Q_i$ by Eq. (23)
7:     end for
8: end for
9: return $D_i$ to clients
```

For local model updates, three optimization objectives are designed: cross-entropy loss, prototype alignment and contrastive prototype loss. Cross-entropy (CE) loss [38] is introduced to represent classification errors. CE loss is calculated by comparing the predictions of the local GNN with the true labels. The formula is as follows:

$$\mathcal{L}_{CE} = -\sum_i y_i \log(softmax(x_i)), \tag{24}$$

where y_i is the one-hot encoding representation of the true labels, and x_i is the model's predicted values. For prototype alignment, each client calculates its local prototypes and aligns them with the domain prototypes computed on the server in Section C. Minimizing the differences between local prototypes and domain prototypes ensures that the client model adapts more accurately to the characteristics of its local data. The formula is as follows:

$$\mathcal{L}_{PA} = \frac{1}{len(J)} \sum_{j\in J} (P_j - D_j)^2, \tag{25}$$

where J represents the number of classes. For contrastive prototype loss, similar to contrastive learning, domain prototypes of the same type are considered positive samples, whereas domain prototypes of different types are considered negative samples. The goal is to reduce the distance between prototypes representing the same domain in local prototypes while increasing the distance to prototypes of different categories. The formula is as follows:

$$\mathcal{L}_{DPC} = \frac{1}{N} \sum_1^N \log\left(1 + \frac{\sum_{D\in\mathcal{N}^k} \exp(sim(z_i, D))}{\sum_{D\in\mathcal{P}^k} \exp(sim(z_i, D))}\right), \tag{26}$$

where N represents the number of sample points, N^k is the set of negative samples, which are the node data from different categories than the current sample, and P^k is the set of positive samples, which are the nodes belonging to the same category as the current sample. $sim(\cdot)$ denotes the similarity computation, z_i represents the node embedding information, and D represents the personalized prototype data of the current client.

In our model, to better balance the contributions of $\mathcal{L}_{PA}$ and $\mathcal{L}_{DPC}$, we introduce two weight parameter ϑ_1 and ϑ_2. The final loss function is defined as follows:

$$\mathcal{L} = \vartheta_1 \mathcal{L}_{PA} + \vartheta_2 \mathcal{L}_{DPC} + \mathcal{L}_{CE}. \tag{27}$$

Through this methodology, the framework addresses the challenges of heterogeneity and data imbalance in FL when applied to graph data, while ensuring data privacy. The next section provides a detailed description of the experimental setup and results, demonstrating the effectiveness and superiority of this framework.

5. Experiments

5.1. Experimental setup

Software and hardware. All experiments were conducted on a Linux server equipped with an NVIDIA RTX 4090 GPU and an Intel Xeon Platinum 8352V CPU, using PyTorch 2.0.0 and Python 3.9. The modeling framework we use is FederatedScope [5].

Datasets. For the experiments, three widely recognized benchmark graph datasets were selected: Cora, CiteSeer, PubMed [39], Amazon-Computers [40] and DBLP-Conference [5], the specifics of each dataset are summarized in Table 1. These datasets vary in scale and domain, making them ideal for training and evaluating graph neural network models. They enable a comprehensive assessment of the robustness and generalization ability of our model. A standard federated graph learning architecture was adopted, and the Louvain [41] algorithm was used to partition these datasets into subgraphs, simulating a distributed learning environment. In

Table 1
Statistics for datasets used for experiments.

Dataset	Nodes	Features	Edges	Classes
Cora	2,708	1,433	5,429	7
CiteSeer	3,327	3,703	4,732	6
PubMed	19,717	500	44,338	3
Computers	13,381	767	245,778	10
DBLP-Conference	52,202	2,302	214,157	4

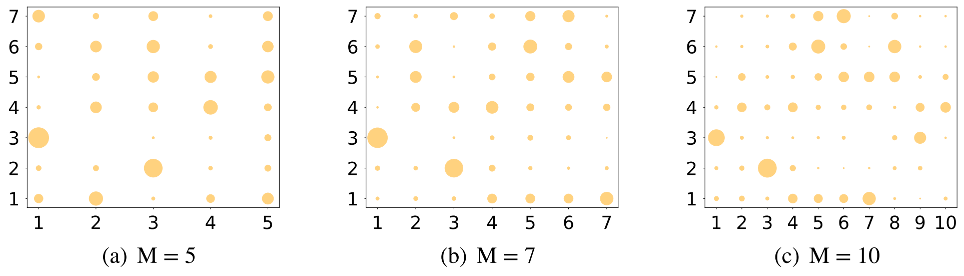


Fig. 4. Distribution of data per client on the Cora dataset where M denotes the total number of clients.

this setup, each client operates on its own subgraph data, training the model locally while synchronizing data and model parameters with a central server. Fig. 4 illustrates the distribution of data on the cora dataset for specific clients.

Baselines and our model. SEFGL was compared with several popular methods for Federated Graph Learning (FGL), including:

- Local: Clients perform local training independently of federated learning.
- FedProto [11]: A baseline model that introduced prototype learning into federated learning.
- FedGH [42]: Uses hypernetworks to generate personalized client models.
- FML-GCN [43]: Utilizes a GCN model framework and improves performance via mutual learning among clients.
- FedKD [13]: Employs knowledge distillation to enable client models to learn a global model, reducing direct transmission of model parameters.
- FPL [9]: Guides local client learning by generating global prototypes.
- FedPCL [44]: A strategy that leverages pre-trained models within a federated learning framework.
- FedAPEN [45]: A framework based on knowledge distillation that adapts to statistical data heterogeneity and supports heterogeneous models across clients.
- FGPL [17]: A heterogeneous federated learning framework incorporating data augmentation and domain prototype calculation.

Model heterogeneity. Model heterogeneity refers to the deployment of models featuring varying core architectures across multiple clients. Specifically, we selected three different GNN architectures (GCN [33], GAT [34], and GPR-GNN [46]) as the foundation for heterogeneous models. These architectures were chosen for their distinct message-passing mechanisms, which improved the diversity of the model. In addition, as classic architectures in the field of GNNs, they provide widely recognized validation for the research framework. To avoid inflated model performance due to overfitting, the three backbone networks were evenly distributed among the clients. To further increase variability among clients using the same backbone network, different numbers of layers and hidden dimensions were configured. Several client counts (m = 5, 7, 10) were examined, with different model configurations assigned to them.

Implementation details. The typical settings for FGL were followed, the datasets were partitioned into subgraphs distributed to distinct clients using the Louvain method, with δ set to 20. Considering the Cora dataset as an example, Fig. 4 illustrates the distribution of the subgraph data nodes held by each client in different client counts. The models were trained and validated in subgraph data specific to each client. During the testing phase, the model was evaluated on the original global graph. The evaluation metrics included accuracy (Acc.), balanced accuracy (bAcc.), and weighted F1 score (F1), providing a comprehensive assessment of model effectiveness. We used SGD as the optimizer, with each method undergoing five local training rounds on the clients. The communication rounds between the clients and the server equaled 150, and the learning rate was set to 0.05. As described in [43], these metrics were calculated by averaging the highest results achieved by each client in all FL rounds.

5.2. *Experimental results*

Main results. We conducted three independent evaluations of the SEFGL model across three distinct datasets. Table 2 presents the average node classification accuracy, balanced accuracy, F1 score, and their standard deviations for each evaluation, providing a comprehensive reflection of the model's classification performance and stability. Compared to all baseline methods, SEFGL exhibited superior performance across all datasets, demonstrating its generalization capability in diverse environments. In scenarios

Table 2
Node classification results. Accuracy, Balanced Accuracy, and Weighted F1 Score on three graph datasets with 5, 7 and 10 clients. The highest value among approaches is marked in bold, while that among baselines is underlined. The metrics for each model are derived from the average of three runs using different random seeds.

	Method	Cora			Citeseer			PubMed		
		M = 5	M = 7	M = 10	M = 5	M = 7	M = 10	M = 5	M = 7	M = 10
Acc.	Global		89.66 ± 1.1			79.27 ± 0.4			87.90 ± 1.4	
	Local	62.46 ± 1.2	56.29 ± 0.8	53.05 ± 1.1	71.74 ± 0.3	64.12 ± 0.7	61.42 ± 1.1	49.21 ± 0.3	50.34 ± 0.3	48.94 ± 1.4
	FedProto	63.34 ± 0.7	56.81 ± 0.7	52.25 ± 0.6	70.50 ± 0.9	64.61 ± 0.6	60.47 ± 0.9	48.93 ± 0.5	42.60 ± 0.4	39.86 ± 1.5
	FedGH	65.29 ± 1.6	56.41 ± 1.2	54.89 ± 3.2	69.62 ± 0.3	63.30 ± 0.2	60.68 ± 0.8	52.05 ± 0.8	49.18 ± 0.3	47.63 ± 0.8
	FML(GCN)	68.93 ± 1.7	59.46 ± 0.5	54.07 ± 2.4	73.82 ± 0.2	66.64 ± 0.4	63.40 ± 0.2	49.28 ± 0.2	37.85 ± 0.5	37.54 ± 0.4
	FedKD	61.46 ± 0.5	56.83 ± 0.5	47.40 ± 0.8	70.95 ± 0.9	63.55 ± 0.3	60.34 ± 0.4	50.88 ± 0.4	35.96 ± 0.8	39.90 ± 0.5
	FPL	63.55 ± 0.5	56.79 ± 0.5	52.17 ± 0.4	68.04 ± 0.8	62.96 ± 0.9	62.47 ± 0.4	50.10 ± 0.5	57.03 ± 0.4	56.13 ± 0.3
	FedPCL	67.83 ± 0.4	60.22 ± 0.8	56.67 ± 1.1	70.80 ± 0.5	66.16 ± 0.9	63.35 ± 0.7	59.18 ± 0.5	60.63 ± 0.4	58.45 ± 0.2
	FedAPEN	84.51 ± 1.4	**84.37 ± 0.8**	83.07 ± 1.9	76.12 ± 0.4	75.85 ± 0.5	74.41 ± 0.6	65.16 ± 0.2	60.62 ± 0.3	59.33 ± 0.3
	FGPL	81.69 ± 0.9	81.34 ± 0.7	81.02 ± 0.9	76.57 ± 0.5	75.23 ± 0.7	73.37 ± 0.8	67.90 ± 0.3	57.90 ± 0.2	68.17 ± 0.6
	SEFGL(ours)	**84.68 ± 0.8**	83.88 ± 0.6	**83.71 ± 1.0**	**76.68 ± 0.3**	**76.01 ± 0.2**	**74.49 ± 0.8**	**82.91 ± 0.2**	**82.14 ± 0.3**	**80.18 ± 0.5**
bAcc.	Global		89.78 ± 1.2			78.64 ± 0.5			85.38 ± 0.6	
	Local	60.09 ± 1.5	46.53 ± 1.0	42.51 ± 0.8	70.51 ± 0.5	63.80 ± 0.2	61.14 ± 0.7	33.74 ± 0.4	34.04 ± 0.4	33.33 ± 0.3
	FedProto	59.97 ± 0.3	48.05 ± 0.7	41.17 ± 0.3	70.41 ± 0.4	64.19 ± 0.6	60.87 ± 0.9	33.32 ± 0.7	33.45 ± 0.9	33.36 ± 0.7
	FedGH	62.51 ± 0.4	48.82 ± 0.2	41.47 ± 2.3	69.22 ± 0.3	62.84 ± 0.4	60.27 ± 1.0	35.60 ± 0.4	33.29 ± 0.3	33.57 ± 0.7
	FML(GCN)	63.41 ± 0.4	50.58 ± 0.5	42.61 ± 0.3	72.92 ± 0.2	66.66 ± 0.6	61.24 ± 0.4	33.32 ± 0.5	33.88 ± 0.4	32.59 ± 0.9
	FedKD	58.24 ± 0.5	48.20 ± 0.8	39.13 ± 0.2	71.62 ± 0.7	65.08 ± 0.2	60.61 ± 0.4	34.80 ± 0.8	33.33 ± 0.3	33.49 ± 0.6
	FPL	58.55 ± 0.5	48.21 ± 0.3	40.89 ± 0.4	68.81 ± 0.6	63.02 ± 0.3	61.12 ± 0.4	34.26 ± 0.3	42.45 ± 0.6	40.34 ± 0.5
	FedPCL	62.73 ± 1.1	50.56 ± 0.2	41.69 ± 0.6	70.22 ± 0.3	64.66 ± 0.5	60.98 ± 0.2	33.33 ± 1.3	31.34 ± 0.6	33.33 ± 0.7
	FedAPEN	60.73 ± 2.1	44.71 ± 1.3	43.39 ± 0.8	71.65 ± 0.5	67.13 ± 0.4	59.81 ± 0.5	58.57 ± 1.2	**51.38 ± 0.3**	45.38 ± 0.2
	FGPL	67.42 ± 0.9	65.96 ± 0.8	66.78 ± 0.8	**72.48 ± 0.8**	65.74 ± 0.9	63.39 ± 0.8	40.23 ± 0.3	33.40 ± 0.5	43.71 ± 0.2
	SEFGL(ours)	**71.61 ± 1.9**	**68.80 ± 1.2**	**68.16 ± 0.6**	71.74 ± 0.4	**68.16 ± 0.3**	**64.78 ± 0.4**	**63.80 ± 0.2**	50.45 ± 0.2	**59.84 ± 0.1**
F1	Global		89.80 ± 0.8			79.18 ± 0.4			86.84 ± 0.7	
	Local	60.14 ± 1.2	51.23 ± 1.5	47.49 ± 0.5	72.21 ± 0.5	65.26 ± 0.6	60.41 ± 0.5	32.56 ± 0.8	35.96 ± 0.6	32.16 ± 0.8
	FedProto	60.69 ± 0.4	52.13 ± 1.1	46.82 ± 0.5	72.28 ± 0.5	65.69 ± 0.4	60.01 ± 0.5	32.16 ± 0.4	30.14 ± 0.2	26.25 ± 0.7
	FedGH	64.20 ± 2.5	54.13 ± 2.2	48.75 ± 2.0	71.46 ± 0.5	64.49 ± 0.2	61.26 ± 0.5	41.08 ± 0.5	32.57 ± 0.6	35.22 ± 0.3
	FML(GCN)	66.01 ± 1.0	52.92 ± 0.9	49.40 ± 0.3	74.78 ± 0.2	67.32 ± 0.4	62.76 ± 1.0	32.55 ± 1.1	30.29 ± 1.4	25.39 ± 3.1
	FedKD	57.60 ± 0.9	51.33 ± 0.3	39.98 ± 1.0	73.32 ± 0.4	64.66 ± 0.4	59.25 ± 0.6	36.91 ± 0.9	23.54 ± 1.3	27.09 ± 1.6
	FPL	60.77 ± 1.3	54.15 ± 1.7	47.72 ± 0.7	72.06 ± 0.3	65.06 ± 1.6	59.75 ± 0.9	34.44 ± 0.3	45.06 ± 0.2	44.78 ± 0.3
	FedPCL	60.57 ± 2.5	51.72 ± 1.5	49.21 ± 1.5	72.97 ± 2.7	65.05 ± 0.6	60.74 ± 0.6	32.16 ± 2.4	24.35 ± 3.2	22.76 ± 2.5
	FedAPEN	60.31 ± 0.5	49.53 ± 1.2	47.48 ± 0.7	71.53 ± 0.5	66.28 ± 0.4	57.52 ± 0.5	61.01 ± 2.1	44.91 ± 1.9	49.50 ± 1.7
	FGPL	80.43 ± 0.9	78.74 ± 1.7	80.33 ± 0.8	75.66 ± 0.8	**75.94 ± 0.7**	73.32 ± 0.9	56.39 ± 4.2	49.55 ± 3.3	60.85 ± 4.1
	SEFGL(ours)	**83.33 ± 0.3**	**81.38 ± 1.3**	**81.26 ± 0.1**	**76.84 ± 0.7**	75.22 ± 0.1	**73.64 ± 0.5**	**79.52 ± 1.3**	**78.19 ± 1.1**	**77.56 ± 2.3**

Table 3
Ablation study of the three components of our framework on the Cora and CiteSeer datasets.

GSE	SGR	PP	Cora			Citeseer		
			M = 5	M = 7	M = 10	M = 5	M = 7	M = 10
×	×	✓	77.41	77.66	80.34	75.73	74.83	73.28
×	✓	×	77.50	77.90	80.09	75.05	75.02	74.35
×	✓	✓	77.67	76.70	79.73	74.66	73.75	73.65
✓	×	×	77.98	77.50	79.09	74.68	75.28	73.07
✓	×	✓	83.66	81.95	82.64	75.09	74.72	74.02
✓	✓	×	82.64	81.43	82.81	75.40	75.34	72.98
SEFGL(ours)			**84.68**	**83.88**	**83.71**	**76.67**	**76.01**	**74.49**

involving varying numbers of clients and degrees of data heterogeneity, SEFGL retained a strong competitive edge. Notably, in the PubMed dataset, SEFGL significantly outperformed other methods, underscoring its advantages in handling highly imbalanced classes and large-scale node environments, thereby further validating the effectiveness of the structural entropy and prototype learning approaches. Our code is publicly available at https://github.com/Jasonxx4/SEFGL.

Ablation study. To verify the effectiveness of the modules in the proposed method, key modules were progressively removed or replaced within the framework. This approach allowed us to quantify the impact of each module on model performance and revealed its specific role in handling data and model heterogeneity (e.g., graph structure enhancement, structural entropy-based graph reconstruction, personalized prototypes, and structural entropy weighting). As listed in Table 3, SEFGL outperforms other methods in most cases. Here, we analyze them one by one. (1) -GSE: Removing the GSE module prevents effective improvement of minority-class node representations, leading to a 3.76% average performance drop in imbalanced data scenarios. (2) -SGR: Removing the SGR module reduces the model's ability to filter noise and redundancy, causing a 1.93% accuracy drop on complex graph datasets.

Table 4
Communication costs for all models during each iteration on the Cora dataset within a 150-round federated graph learning setting.

Method	Practice(MBs)	Method	Practice(MBs)
Local	0	FPL	2.59
FedProto	1.91	FedPCL	2.23
FedGH	1.65	FedAPEN	101.18
FML(GCN)	144.53	FGPL	1.95
FedKD	151.22	**SEFGL(ours)**	**4.13**

Table 5
Comparison of model performance under different heterogeneous conditions, the highest value among approaches is marked in bold.

Methods	Cora			Computers			DBLP_Conf		
	HtFE2	HtFE4	HtFE5	HtFE2	HtFE4	HtFE5	HtFE2	HtFE4	HtFE5
FedProto	63.28 ± 1.3	61.52 ± 1.5	59.36 ± 1.8	51.50 ± 3.0	49.95 ± 2.5	45.39 ± 3.1	62.94 ± 1.6	62.67 ± 1.8	62.51 ± 1.5
FedGH	59.13 ± 2.2	54.59 ± 1.8	50.61 ± 2.4	57.94 ± 1.6	57.36 ± 1.5	56.96 ± 1.7	40.60 ± 1.0	40.28 ± 0.9	38.71 ± 1.4
FML	56.38 ± 0.9	53.05 ± 1.1	48.49 ± 1.2	58.20 ± 1.6	57.56 ± 1.0	56.72 ± 1.8	46.26 ± 0.9	45.33 ± 1.1	45.01 ± 1.0
FedKD	59.18 ± 0.6	56.21 ± 1.1	57.23 ± 0.9	50.04 ± 1.4	49.73 ± 1.2	49.39 ± 1.1	51.25 ± 1.0	51.18 ± 1.5	50.77 ± 1.3
FPL	57.37 ± 0.7	57.30 ± 0.8	57.26 ± 0.5	53.83 ± 1.5	52.93 ± 1.9	52.59 ± 1.8	52.73 ± 0.6	51.93 ± 1.0	50.33 ± 0.8
FedPCL	54.28 ± 0.6	52.30 ± 0.8	50.76 ± 0.5	47.20 ± 1.5	47.00 ± 1.1	46.72 ± 1.8	44.34 ± 0.7	43.11 ± 1.2	41.48 ± 1.3
FedAPEN	85.52 ± 1.5	83.19 ± 1.2	82.97 ± 1.0	81.34 ± 1.3	80.64 ± 1.2	80.12 ± 0.9	50.73 ± 1.5	50.35 ± 1.8	49.82 ± 1.4
FGPL	85.20 ± 1.2	84.81 ± 1.7	84.45 ± 0.8	79.04 ± 1.3	78.41 ± 0.9	77.94 ± 1.0	68.34 ± 0.8	67.59 ± 1.2	66.48 ± 1.0
SEFGL	**85.76 ± 1.0**	**85.36 ± 0.8**	**85.26 ± 0.7**	**83.77 ± 1.2**	**83.32 ± 0.9**	**83.13 ± 0.8**	**75.57 ± 1.5**	**74.44 ± 1.2**	**73.81 ± 1.0**

(3) -PP: Removing the PP module limits global information utilization, increasing client model discrepancies and reducing accuracy by up to 2.45% in multi-client scenarios.

These results highlight the critical role of each module in addressing heterogeneity and the importance of their synergy. Future work could explore dynamic parameter adjustments for complex FGL scenarios.

Communication cost. Table 4 depicts our communication costs. Among the models that use knowledge distillation techniques, FML, FedKD, and FedAPEN incur the highest transmission overhead because of the additional model transfer. In contrast, our approach based on prototype learning offers fundamental advantages in transmission efficiency. Although our model's overhead is slightly higher than that of other prototype-based models, it remains significantly lower than that of the knowledge distillation models. In summary, our SEFGL not only achieves superior model accuracy but also maintains efficient communication.

5.3. Diagnostic analysis

5.3.1. Impact of model heterogeneity

We further evaluated SEFGL's performance under three scenarios with increasing model heterogeneity: HtFE2 (containing GCN and GAT), HtFE4 (containing GCN, GAT, GIN and GPR), and HtFE5 (containing GCN, GAT, GIN, GPR and SAGE). All models were configured with 2-layer architectures and hidden dimensions of 64. Table 5 lists the test accuracy results. Experimental results demonstrate that nearly all baseline methods exhibited performance degradation as model heterogeneity increased, showing an average performance decline of 2.31% from HtFE2 to HtFE5. In contrast, SEFGL consistently achieved superior performance across all scenarios. It not only maintained the highest classification accuracy (surpassing the best baseline by 7.33%), but also demonstrated significantly better robustness with only 1.13% average performance degradation. These findings conclusively validate SEFGL's effectiveness in handling heterogeneous model integration scenarios.

5.3.2. Impact of local training rounds

In our study using the Cora dataset with 10 heterogeneous clients, we thoroughly examined the effect of varying local client update rounds on the accuracy of the FL model. As illustrated in the Table 6, we can observe that the number of local training epochs (E) significantly impacts model performance. When the number of local training rounds is extremely low, the model may suffer from underfitting and fail to reach optimal convergence. Conversely, if the number of rounds is exceedingly high, overfitting may occur, leading to a decline in performance. However, our proposed SEFGL framework consistently outperformed the baseline models across various settings and substantially outperformed the second-best model in most cases. This result clearly demonstrates the flexibility and robustness of the SEFGL framework in adapting to different training conditions.

5.3.3. Impact of the degree of data heterogeneity

In this section, we will examine the impact of the partition parameter delta in the Louvain algorithm on model performance. The delta value quantifies the degree of variation in community partitioning between clients, specifically reflecting the difference in node counts. Through systematic experiments, we found that delta exerts a direct and significant influence on model performance. Fig. 5 illustrates how model performance declines with increasing delta increases, indicating that greater inter-client differences pose

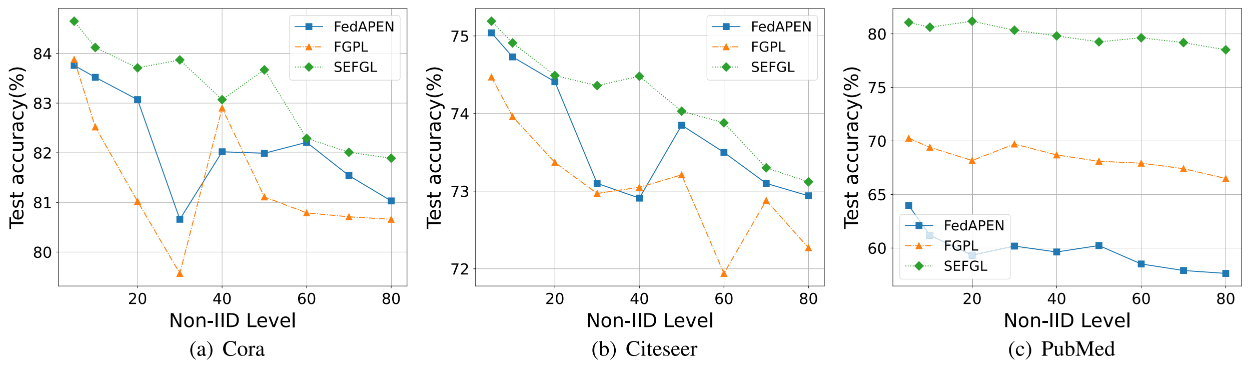


Fig. 5. The impact of data heterogeneity induced by varying δ values on model performance under Louvain community partitioning in the Cora, Citeseer and PubMed dataset.

Table 6
The effect of varying local update epochs on model performance in the Cora dataset. The highest value among approaches is marked in bold, while that among baselines is underlined.

Method	local update epochs			
	E = 3	E = 5	E = 7	E = 10
Local	52.44	53.05	52.91	52.12
FedProto	52.12	52.25	52.01	51.38
FedGH	54.23	54.89	54.16	53.96
FML(GCN)	53.55	54.07	53.61	53.42
FedKD	47.04	47.40	47.31	47.14
FPL	52.42	52.17	52.08	51.95
FedPCL	56.88	56.67	56.41	56.12
FGPL	80.97	81.02	81.14	80.91
FedAPEN	<u>82.49</u>	<u>83.07</u>	<u>82.97</u>	<u>82.77</u>
SEFGL(ours)	**82.56**	**83.71**	**83.25**	**82.84**

Table 7
On Cora, Citeseer and PubMed datasets, the comparison with the model isomorphism approach.

Method	Cora			Citeseer			PubMed		
	m = 5	m = 7	m = 10	m = 5	m = 7	m = 10	m = 5	m = 7	m = 10
Global		87.78			76.91			88.38	
Local	61.54	45.32	32.42	73.85	62.87	48.91	83.81	72.34	59.19
FedAvg [47]	86.63	86.21	86.01	76.37	76.57	75.92	85.29	84.27	84.57
FedProx [48]	86.6	86.27	86.22	77.15	77.28	76.87	85.21	84.01	84.98
FedOpt [49]	86.11	85.89	85.2	76.96	76.82	76.71	84.39	84.1	83.91
FedSage [20]	86.86	86.59	86.32	77.91	77.82	77.3	87.75	87.51	87.49
FGSSL [50]	88.34	88.56	88.01	80.43	80.21	80.01	88.25	87.75	87.60
SEFGL(ours)	86.18	86.42	86.06	76.90	76.58	76.39	84.07	84.18	84.12

challenges in optimizing modularity and complicate the identification of natural community structures in the network. Nevertheless, our proposed SEFGL framework consistently outperforms baseline models across various degrees of data heterogeneity, demonstrating robust adaptability to heterogeneous data conditions.

5.3.4. *Performance analysis under model isomorphism*

We designed experiments to compare the performance of heterogeneous models with traditional homogeneous models. We used the FGSSL [50] model as the baseline for comparison. To ensure fairness, we set each client's model to be a two-layer GAT with a hidden layer size of 128 and trained it for 200 rounds while maintaining all other settings unchanged. As shown in Table 7, even under conditions of model homogeneity, our SEFGL framework demonstrates robust performance.

5.3.5. *Impact of the noise scales*

We conducted an evaluation of SEFGL's noise resistance on the Cora dataset under a 10-client configuration, with the accuracy results depicted in Fig. 7. We applied Uniform Noise to the dataset, systematically testing noise rates of $\{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$.

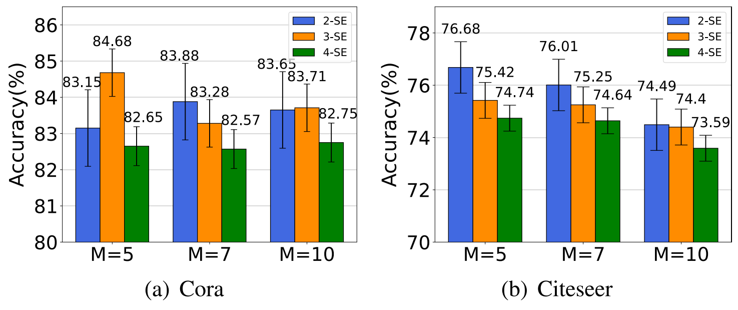


Fig. 6. Results of SEFGL on the Cora and CiteSeer datasets with different encoding tree heights.

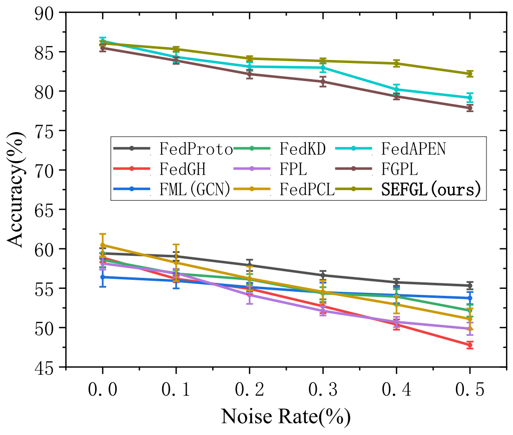


Fig. 7. Evaluate the performance of different algorithms under varying levels of noise interference.

Although increased noise levels generally resulted in reduced accuracy across all methods, SEFGL demonstrated superior performance in noise resistance, achieving an average accuracy improvement of over 20% compared to other methods. This finding highlights SEFGL's robustness and scalability in complex FL environments, underscoring its potential for deployment in challenging distributed learning scenarios.

5.4. *Hyperparameter analysis*

5.4.1. *Impact of the encoding tree height*

Regarding the influence of the encoding tree height H, we evaluated two different datasets and scenarios with varying numbers of participating clients. As shown in Fig. 6, the optimal tree height differs across datasets and numbers of participating clients, where n-SE denotes the dimension of the structural entropy, also approximated by the height of the coding tree. For example, in the case of the Cora dataset, the optimal tree height H is 2 when the number of clients is 7, while it is 3 when the number of clients is 5 or 10. In contrast, for the CiteSeer dataset, the optimal results are attained with a tree height of 2. Therefore, the correlation between encoding tree height and model performance warrants further investigation in future work.

5.4.2. *Impact of client aggregation numbers*

To assess the impact of aggregating different numbers of clients by a single client per round, we evaluated scenarios with 10 participating clients under two distinct datasets. We set the number of aggregated clients to 3, 5, 7, and 10. Fig. 8 illustrates that optimal performance occurs when seven clients are aggregated for both datasets. The Cora dataset shows a significant variance in performance with different numbers of aggregated clients, whereas the CiteSeer dataset demonstrates relative stability.

5.4.3. *Impact of discount factor*

We conducted an in-depth study on the impact of the discount factor ω_v. Experimental results in Fig. 9 demonstrate that the fixed discount factor significantly influences model performance. Specifically, when the discount factor is relatively small (e.g. 0.1 to 0.3), the model exhibits lower performance owing to insufficient enhancement effects from virtual edges. Conversely, with larger discount factors (e.g. 0.7 to 0.8), model performance begins to decline, likely attributable to excessive noise introduction. Experimental validation reveals that the optimal value for the fixed discount factor lies between 0.5 and 0.6, where the model achieves a balance between enhancement effectiveness and noise interference and attains peak performance. However, in comparison, a dynamic discount factor demonstrates superior adaptability by automatically adjusting according to variations in nodes' risk scores and other parameters.

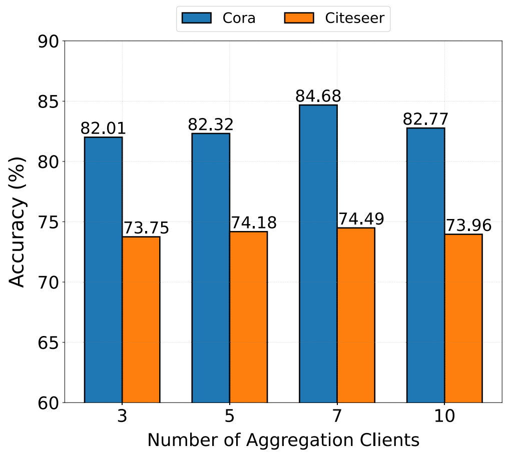


Fig. 8. Results of the number of clients aggregated by each client affects model performance on the Cora and Citeseer datasets.

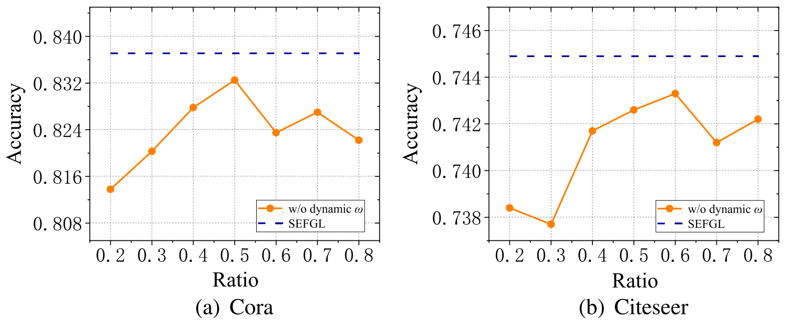


Fig. 9. Results of SEFGL on the Cora and CiteSeer datasets with different discount factor.

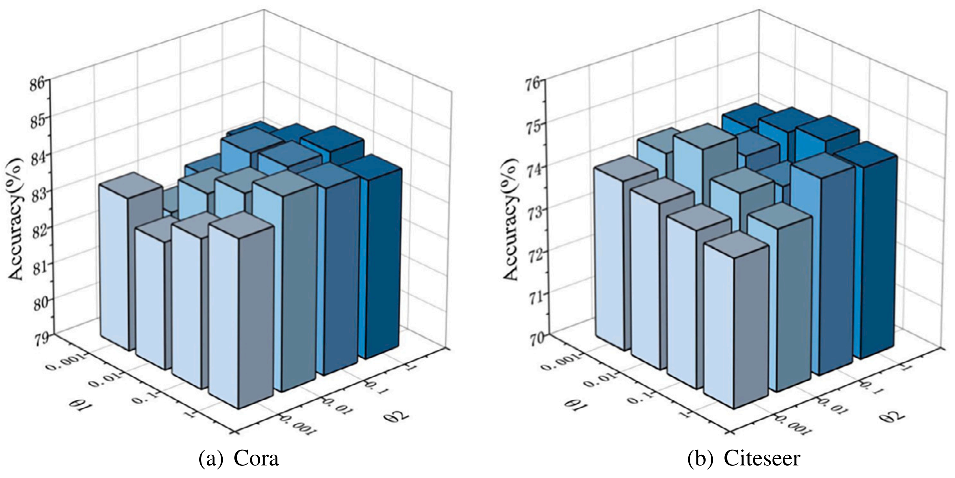


Fig. 10. Sensitive analysis for two parameters ϑ_1 and ϑ_2.

This dynamic adjustment mechanism enables sustained optimal performance across different training phases and data distributions, proving particularly advantageous when handling complex data patterns and highly heterogeneous scenarios.

5.4.4. Sensitivity analysis of the loss function

We conducted a systematic study of the hyperparameters ϑ_1 and ϑ_2 and evaluated the performance of SEFGL across 10 clients on the Cora and Citeseer datasets. Specifically, we adjusted the weight parameters ϑ_1 and ϑ_2 within the range $\{1e-3, 1e-2, 1e-1, 1\}$ to comprehensively analyze their impact on model performance. The results of the sensitivity analysis are shown in Fig. 10. The experiments demonstrated that the performance fluctuation of SEFGL across different hyperparameter combinations remained within

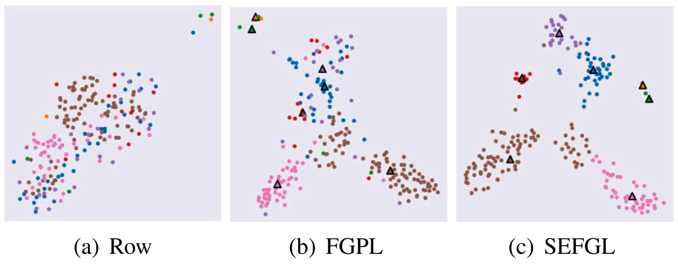


Fig. 11. A visualization displays the one client's local prototype node embeddings on the Cora dataset, using various colors to denote different data categories. Triangles represent the personalized prototypes.

2%, indicating that SEFGL exhibits strong robustness to variations in ϑ_1 and ϑ_2. This suggests that the model's performance is not highly sensitive to the selection of these two hyperparameters.

5.5. Visualization

To evaluate the effectiveness of our approach, a t-SNE-based visualization analysis of single-client node embeddings was conducted. Fig. 11 (a) displays the original node embeddings, where each point's color represents its corresponding category. Fig. 11 (b) illustrates the node embedding distribution generated by the FGPL method. In comparison to Fig. 11 (a), nodes from different categories form more distinct clusters. However, overlapping regions persist between some categories, suggesting limitations in handling category boundaries with existing methods. Fig. 11 (c) presents node embeddings produced by the proposed SEFGL method. In this visualization, nodes of different categories form clearer and more compact clusters, with distinct category boundaries and significantly reduced overlapping regions. Furthermore, the SEFGL method achieves superior separation performance for hard-to-distinguish categories. The visualization results validate the effectiveness of the SEFGL method in enhancing category separability and demonstrate its robustness in managing complex data distributions.

6. Conclusion

In this study, we designed and implemented a novel FGL framework, named SEFGL, to address the challenges of data heterogeneity and model heterogeneity. Within the SEFGL framework, we employed two key components to address these challenges. On one hand, we utilized a structure entropy-based graph learning approach to improve adaptability to diverse model distributions and alleviate data heterogeneity. On the other hand, we incorporated a structure entropy-based prototype learning method, in which global information is transmitted to participating clients through prototype data to mitigate model heterogeneity. This work represents the first application of structure entropy and prototype learning in FGL, achieving significant advances in model personalization and privacy protection while effectively addressing data and model heterogeneity challenges. Experiments conducted on five real-world datasets demonstrate that SEFGL significantly outperforms baseline models with reduced communication costs.

Future work will involve extending the SEFGL framework to a broader range of graph learning tasks, such as graph classification and link prediction. The application of SEFGL to these tasks is expected to further demonstrate its versatility and effectiveness in addressing diverse challenges in FGL. Additionally, we will explore the adaptability of SEFGL in real-world scenarios with dynamic graph structures, investigate adaptive clustering mechanisms, and optimize algorithms to enhance computational efficiency, thereby further strengthening its potential for practical applications.

CRediT authorship contribution statement

Zhehao Dai: Writing – original draft, Methodology, Data curation, Conceptualization. **Guojiang Shen:** Validation. **Haopeng Yuan:** Investigation. **Shangfei Zheng:** Writing – review & editing. **Yuyue Hu:** Visualization. **Jiaxin Du:** Writing – review & editing. **Xiangjie Kong:** Writing – review & editing. **Feng Xia:** Writing – review & editing, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62476247, 62072409 and 62402442, in part by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang under Grant 2024C01214, and in part by the Zhejiang Provincial Natural Science Foundation under Grant LR21F020003 and LQ24F020038.

Data availability

Data will be made available on request.

References

[1] Lingyun Wang, Hanlin Zhou, Yinwei Bao, Xiaoran Yan, Guojiang Shen, Xiangjie Kong, Horizontal federated recommender system: a survey, ACM Comput. Surv. 56 (9) (2024) 1–42.

[2] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, S. Yu Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2020) 4–24.

[3] Qiang Yang, Yang Liu, Tianjian Chen, Yongxin Tong, Federated machine learning, ACM Trans. Intell. Syst. Technol. (Mar 2019) 1–19.

[4] Xingbo Fu, Binchi Zhang, Yushun Dong, Chen Chen, Jundong Li, Federated graph machine learning: a survey of concepts, techniques, and applications, SIGKDD Explor. 24 (2) (2022) 32–47.

[5] Zhen Wang, Weirui Kuang, Yuexiang Xie, Liuyi Yao, Yaliang Li, Bolin Ding, Jingren Zhou, Federatedscope-gnn: towards a unified, comprehensive and efficient package for federated graph learning, in: SIGKDD, 2022, pp. 4110–4120.

[6] Matias Mendieta, Taojiannan Yang, Pu Wang, Minwoo Lee, Zhengming Ding, Chen Chen, Local learning matters: rethinking data heterogeneity in federated learning, in: CVPR, 2022, pp. 8397–8406.

[7] Mang Ye, Xiuwen Fang, Bo Du, Pong C. Yuen, Dacheng Tao, Heterogeneous federated learning: state-of-the-art and research challenges, ACM Comput. Surv. 56 (3) (2023) 1–44.

[8] Zhining Liu, Ruizhong Qiu, Zhichen Zeng, Hyunsik Yoo, David Zhou, Zhe Xu, Yada Zhu, Kommy Weldemariam, Jingrui He, Hanghang Tong, Class-imbalanced graph learning without class rebalancing, in: ICML, 2024.

[9] Wenke Huang, Mang Ye, Zekun Shi, He Li, Bo Du, Rethinking federated learning with domain shift: a prototype view, in: CVPR, IEEE, 2023, pp. 16312–16322.

[10] Angsheng Li, Yicheng Pan, Structural information and dynamical complexity of networks, IEEE Trans. Inf. Theory 62 (6) (2016) 3290–3339.

[11] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, Chengqi Zhang, Fedproto: federated prototype learning across heterogeneous clients, in: AAAI, vol. 36, 2022, pp. 8432–8440.

[12] Rui Liu, Pengwei Xing, Zichao Deng, Anran Li, Cuntai Guan, Han Yu, Federated graph neural networks: overview, techniques, and challenges, IEEE Trans. Neural Netw. Learn. Syst. (2024).

[13] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, Xing Xie, Communication-efficient federated learning via knowledge distillation, Nat. Commun. 13 (1) (2022) 2032.

[14] Xingbo Fu, Zihan Chen, Binchi Zhang, Chen Chen, Jundong Li, Federated graph learning with structure proxy alignment, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 827–838.

[15] Zihan Tan, Guancheng Wan, Wenke Huang, Mang Ye, FedSSP: federated graph learning with spectral knowledge and personalized preference, in: The Thirty-Eighth Annual Conference on Neural Information Processing Systems, 2024.

[16] Xingbo Fu, Zihan Chen, Yinhan He, Song Wang, Binchi Zhang, Chen Chen, Jundong Li, Virtual nodes can help: tackling distribution shifts in federated graph learning, in: AAAI, 2025.

[17] Xiangjie Kong, Haopeng Yuan, Guojiang Shen, Hanlin Zhou, Weiyao Liu, Yao Yang, Mitigating data imbalance and generating better prototypes in heterogeneous federated graph learning, Knowl.-Based Syst. 296 (2024) 111876.

[18] Shiyu Wang, Jiahao Xie, Mingming Lu, Neal N. Xiong, Fedgraph-kd: an effective federated graph learning scheme based on knowledge distillation, in: BigDataSecurity, HPSC, IDS, IEEE, 2023, pp. 130–134.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144.

[20] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, Siu Ming Yiu, Subgraph federated learning with missing neighbor generation, NeurIPS 34 (2021) 6671–6682.

[21] Xiangjie Kong, Wenyi Zhang, Hui Wang, Mingliang Hou, Xin Chen, Xiaoran Yan, Sajal K. Das, Federated graph anomaly detection via contrastive self-supervised learning, IEEE Trans. Neural Netw. Learn. Syst. (2024).

[22] Chunnan Wang, Bozhou Chen, Geng Li, Hongzhi Wang, Automated graph neural network search under federated learning framework, IEEE Trans. Knowl. Data Eng. 35 (10) (2023) 9959–9972.

[23] Junran Wu, Xueyuan Chen, Bowen Shi, Shangzhe Li, Ke Xu, Sega: structural entropy guided anchor view for graph contrastive learning, in: ICML, PMLR, 2023, pp. 37293–37312.

[24] Junran Wu, Xueyuan Chen, Ke Xu, Shangzhe Li, Structural entropy guided graph hierarchical pooling, in: ICML, PMLR, 2022, pp. 24017–24030.

[25] Dongcheng Zou, Hao Peng, Xiang Huang, Renyu Yang, Jianxin Li, Jia Wu, Chunyang Liu, Philip S. Yu, Se-gsl: a general and effective graph structure learning framework through structural entropy optimization, in: WWW, 2023, pp. 499–510.

[26] Mingyu Zhao, Weidong Yang, Feiping Nie, Auto-weighted orthogonal and nonnegative graph reconstruction for multi-view clustering, Inf. Sci. 632 (2023) 324–339.

[27] Zhihao Peng, Hui Liu, Yuheng Jia, Junhui Hou, Egrc-net: embedding-induced graph refinement clustering network, IEEE Trans. Image Process. 32 (2023) 6457–6468.

[28] Xiaojun Yang, Weihao Zhao, Jing Wang, Siyuan Peng, Feiping Nie, Auto-weighted graph reconstruction for efficient ensemble clustering, Inf. Sci. 689 (2025) 121486.

[29] Jake Snell, Kevin Swersky, Richard Zemel, Prototypical networks for few-shot learning, NeurIPS 30 (2017).

[30] Xiyue Gao, Zhuoqi Ma, Jiangtao Cui, Xiaofang Xia, Cai Xu, Hierarchical category-enhanced prototype learning for imbalanced temporal recommendation, in: ACM MM, 2023, pp. 6181–6189.

[31] Weiyi Yang, Richong Zhang, Junfan Chen, Lihong Wang, Jaein Kim, Prototype-guided pseudo labeling for semi-supervised text classification, in: ACL, 2023, pp. 16369–16382.

[32] Jianqing Zhang, Yang Liu, Yang Hua, Jian Cao, Fedtgp: trainable global prototypes with adaptive-margin-enhanced contrastive learning for data and model heterogeneity in federated learning, in: AAAI, vol. 38, 2024, pp. 16768–16776.

[33] Thomas N. Kipf, Max Welling, Semi-supervised classification with graph convolutional networks, in: ICLR, 2017.

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio, Graph attention networks, in: ICLR, 2018.

[35] Tianxiang Zhao, Xiang Zhang, Suhang Wang, Graphsmote: imbalanced node classification on graphs with graph neural networks, in: WSDM, 2021, pp. 833–841.

[36] Liang Duan, Xiang Chen, Wenjie Liu, Daliang Liu, Kun Yue, Angsheng Li, Structural entropy based graph structure learning for node classification, Proc. AAAI Conf. Artif. Intell. 38 (2024) 8372–8379.

[37] Saquib Sarfraz, Vivek Sharma, Rainer Stiefelhagen, Efficient parameter-free clustering using first neighbor relations, in: CVPR, 2019, pp. 8934–8943.

[38] Zhilu Zhang, Mert Sabuncu, Generalized cross entropy loss for training deep neural networks with noisy labels, NeurIPS 31 (2018).

[39] Zhilin Yang, William Cohen, Ruslan Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in: ICML, PMLR, 2016, pp. 40–48.

[40] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, Stephan Günnemann, Pitfalls of graph neural network evaluation, arXiv preprint arXiv:1811.05868, 2018.

[41] Xinyu Que, Fabio Checconi, Fabrizio Petrini, John A. Gunnels, Scalable community detection with the Louvain algorithm, in: IPDPS, IEEE, 2015, pp. 28–37.
[42] Liping Yi, Gang Wang, Xiaoguang Liu, Zhuan Shi, Han Yu, Fedgh: heterogeneous federated learning with generalized global header, in: ACM MM, 2023, pp. 8686–8696.
[43] Tao Shen, Jie Zhang, Xinkang Jia, Fengda Zhang, Gang Huang, Pan Zhou, Kun Kuang, Fei Wu, Chao Wu, Federated mutual learning, arXiv preprint arXiv:2006.16765, 2020.
[44] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, Jing Jiang, Federated learning from pre-trained models: a contrastive learning approach, NeurIPS 35 (2022) 19332–19344.
[45] Zhen Qin, Shuiguang Deng, Mingyu Zhao, Xueqiang Yan, Fedapen: personalized cross-silo federated learning with adaptability to statistical heterogeneity, in: SIGKDD, 2023, pp. 1954–1964.
[46] Eli Chien, Jianhao Peng, Pan Li, Olgica Milenkovic, Adaptive universal generalized pagerank graph neural network, in: ICLR, 2021.
[47] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, Zhihua Zhang, On the convergence of fedavg on non-iid data, in: ICLR, 2020.
[48] Xiaotong Yuan, Ping Li, On convergence of fedprox: local dissimilarity invariant bounds, non-smoothness and beyond, NeurIPS 35 (2022) 10752–10765.
[49] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, Hugh Brendan McMahan, Adaptive federated optimization, in: ICLR, 2021.
[50] Wenke Huang, Guancheng Wan, Mang Ye, Bo Du, Federated graph semantic and structural learning, in: Edith Elkind (Ed.), Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23, in: International Joint Conferences on Artificial Intelligence Organization, vol. 8, 2023, pp. 3830–3838, Main Track.